CS-Exam

Mohsen Pirmoradiyan

3/24/2020

I solemnly swear upon my honour that I wrote the exam honestly, I did not use any unpermitted aids, nor did I communicate with anybody except of the course examiners.

Mohsen Pirmoradiyan

Assignment 1

Q1.1

```
target = function(x, c){
  f = c * sqrt(2*pi) * exp(-c^2/(2*x)) * x^(-3/2)
  return(f)
}

propose = function(x, alpha, Tmin){
  g = ((alpha-1)/Tmin) * (x/Tmin)^(-alpha)
  return(g)
}

x = seq(0.1,10, 0.5)
f = target(x, c=0.5)

g = propose(x, alpha = 1.4, 0.1)
h = f/g

#plot(x,f, main = "target", col="red")
#lines(x, g, main = "Propose x=(0,100)", col="blue")


df = data.frame(x=x, target=f, majorize=g, ratio = h)
ggplot(df)+geom_line(mapping = aes(x,f, color="target"))+
  geom_line(mapping = aes(x,g, color="powerLaw"))
```

Q. 1. 1

Target $f(x) = c\sqrt{2\pi}\, e^{-c^2/(2x)}\, x^{(-\frac{3}{2})}$ $x > 0$

proposal: $g(x) = \frac{\alpha - 1}{T_{min}} \left(\frac{x}{T_{min}} \right)^{-\alpha}$ $x > T_{min}$

$$h(x) = \frac{f(x)}{g(x)} = \frac{c\sqrt{2\pi}\, e^{\frac{-c^2}{2x}}\, x^{-3/2}}{\frac{\alpha-1}{T_{min}} \left(\frac{x}{T_{min}} \right)^{-\alpha}} = \underbrace{\frac{c\sqrt{2\pi}\, T_{min}^{(1-\alpha)}}{\alpha - 1}}_{k} \left(e^{\frac{-c^2}{2x}} \right) x^{-\frac{3}{2}+\alpha}$$

$$\Rightarrow h(x) = k\, e^{\frac{-c^2}{2x}}\, x^{-\frac{3}{2}+\alpha} \leq M$$

$$\frac{dh}{dx} = k\, e^{\frac{-c^2}{2x}}\, x^{-\frac{3}{2}+\alpha} \left[\frac{c^2}{2x^2} + \alpha - \frac{5}{2} \right] = 0$$

$$\Rightarrow \frac{c^2}{2x^2} + \alpha - \frac{5}{2} = 0 \Rightarrow x^2 = \frac{c^2}{5 - 2\alpha}$$

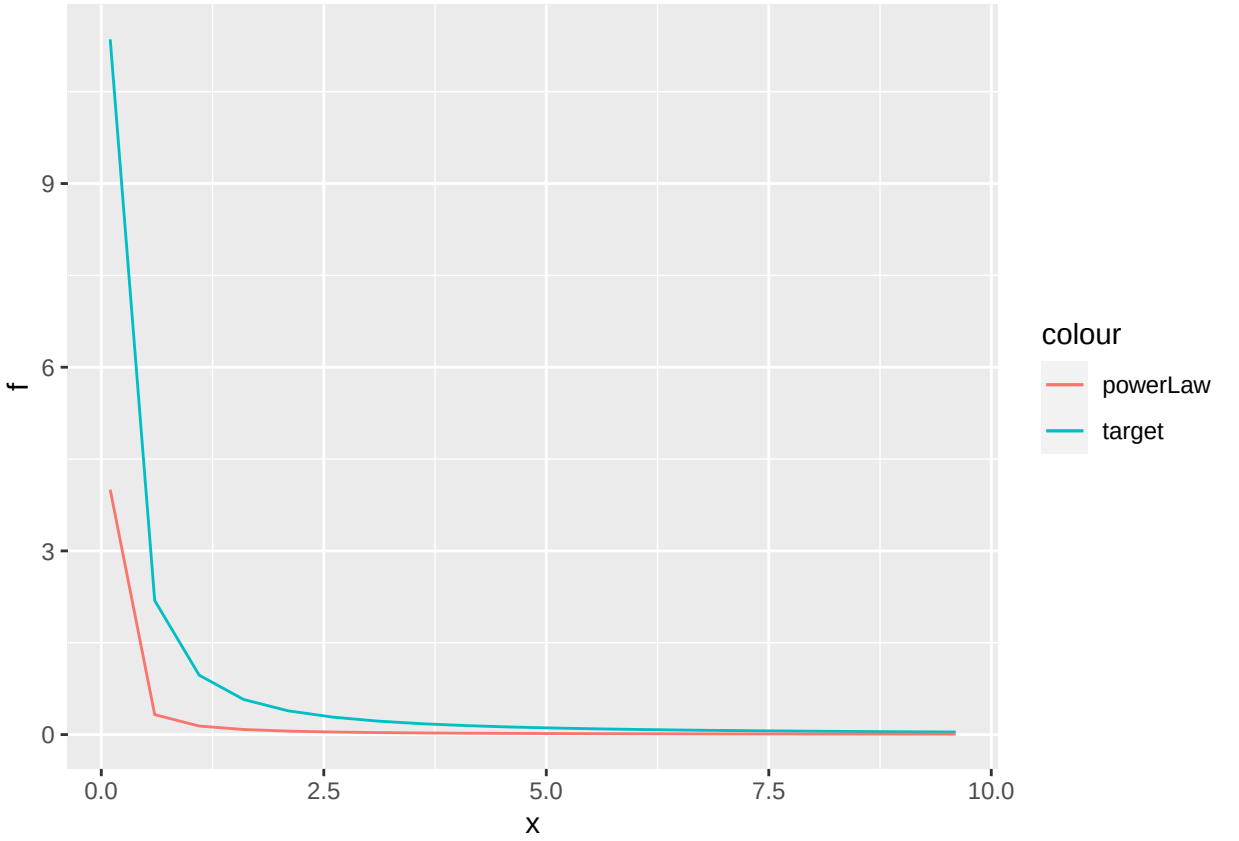
~~if~~ ~~and~~ $5 - 2\alpha > 0 \Rightarrow 5 > 2\alpha \Rightarrow \alpha < \frac{5}{2}$

$$\Rightarrow 1 < \alpha < 5/2$$

$T_{min} = 0.1$, $c = 0.5$, $\alpha = 1.4$

$$\Rightarrow x^2 = \frac{0.5^2}{5 - 2.8} = 0.113636 \Rightarrow \underline{x = 0.3371}$$

$$\Rightarrow h(x = 0.3371) = \frac{f}{g} = 6.05 \rightarrow M$$




```
  #geom_line(mapping = aes(x,h, color="ratio"))
```

Depending on Tmin, as the plots show propose function is not supported at X < Tmin. In this region it can not be used to sample from target function. we can sample the target for x>Tmin by using power law distribution. By trying different values for parameters, I decided to use Tmin = 0.1, c = 0.5, and alpha = 1.4(alpha should be <1.5(in handwritten solution))

```
f = target(x = 0.3371, c = 0.5)
g = propose(x=0.3371, alpha= 1.4, Tmin=0.1)
h = f/g
M=h
```

M is not a good value for my choices. M should be close to 1 as much as possible

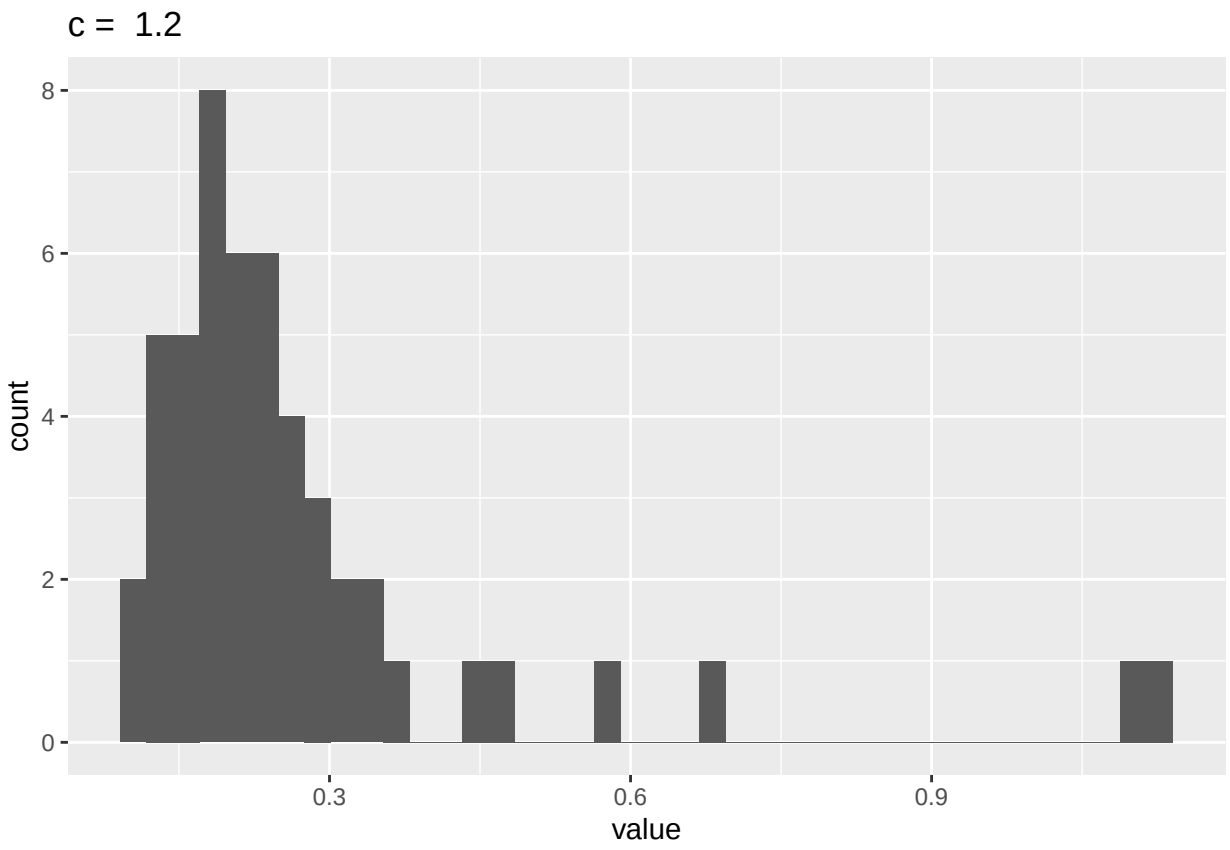
Q1.2

```
myAccept = function(size, c=0.5){
R=0
Y=vector(length = size)
for (i in 1:size) {
repeat {
y = poweRlaw::rplcon(1, xmin=0.1, alph=1.4)
h = target(y, c=c)/M*propose(y, alpha = 1.4, Tmin=0.1)
U = runif(1)
if(U <= h){Y[i]=y; break}
else{R = R+1}
}
}
return(list(Y=Y, Reject=R))
}


histFunc = function(c, size, bins){
  z1 = myAccept(size, c)$Y
  cat("Mean : ", mean(z1),"\n")
  cat("Variance:", var(z1),"\n")

  df = data.frame(value =z1)
  ggplot(df, aes(value))+geom_histogram(bins = bins)+

    labs(title = paste('c = ',as.character(c)))


}
```

Q1.3

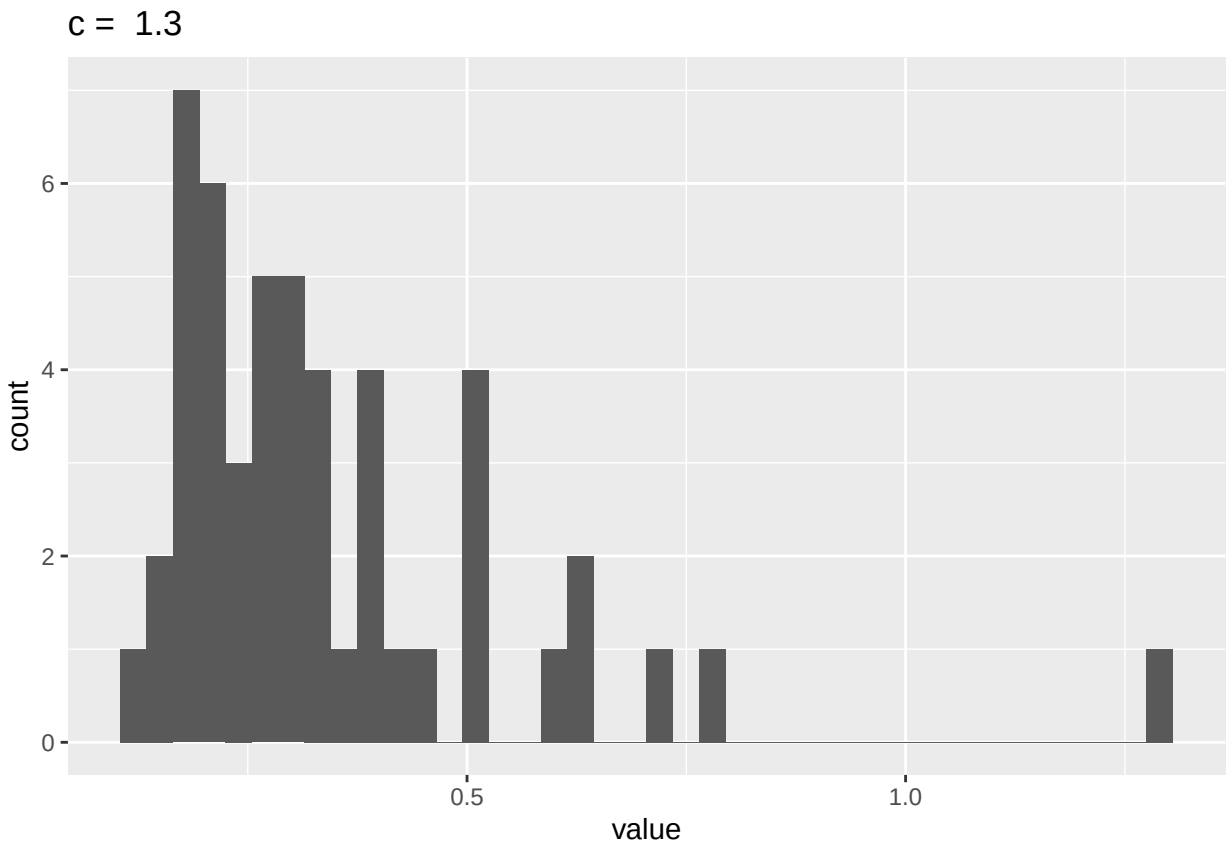
```
histFunc(c=1.2, 50, 40)
```

```
## Mean :  0.2766323
## Variance: 0.0429995
```



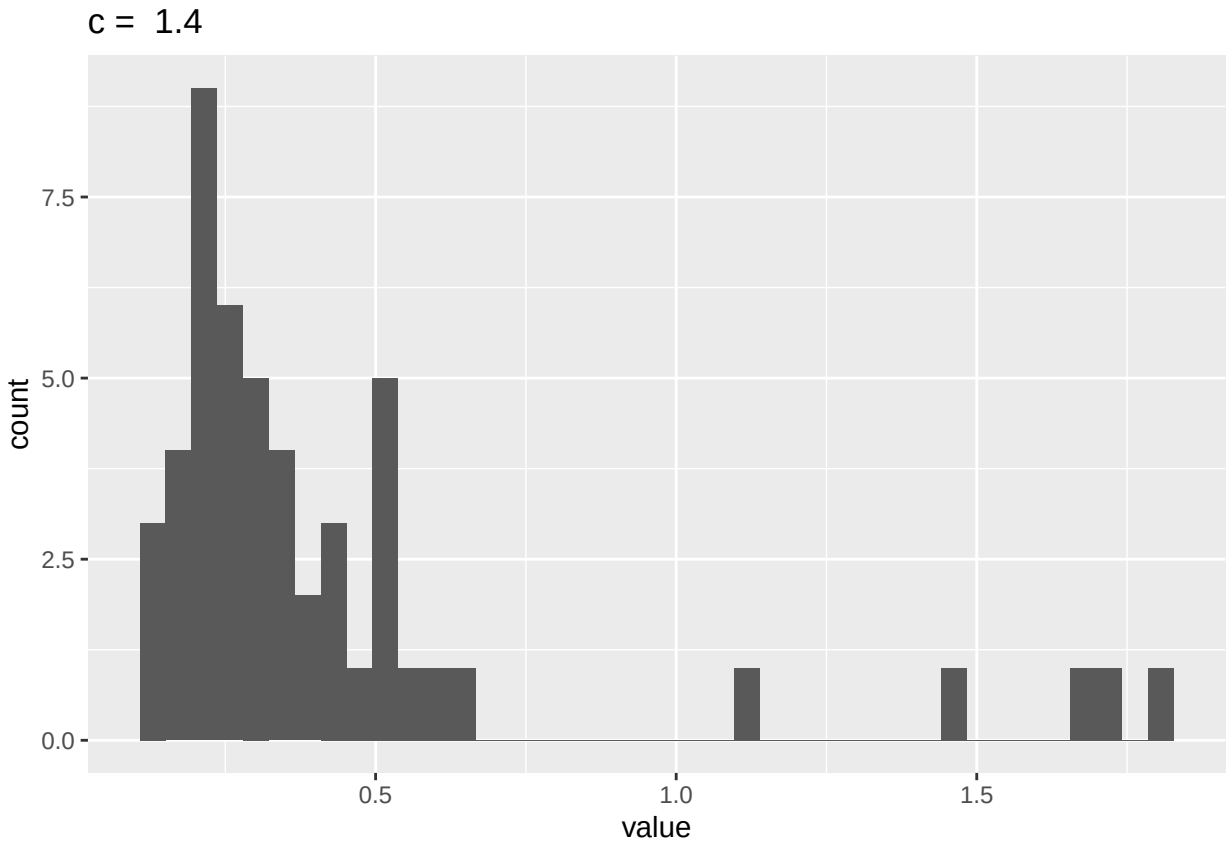

```
histFunc(c=1.3, 50, 40)
```

```
## Mean :  0.3478223
## Variance: 0.04164477
```




```
histFunc(c=1.4, 50, 40)
```

```
## Mean :  0.4408325
## Variance: 0.1631623
```



As c increase the mean increases. Variance also seems to increases.

The obtaining samples do not look like good. The problem might be due to inappropriate choices of parameters

Assignment 2

Q2.1

```
f = function(x1, x2){
  f=x1^2 + 0.5*x2^2 + 3
  return(f)
}
```

Q 2.1

$$f(x_1, x_2) = x_1^2 + \frac{1}{2}x_2^2 + 3$$

$$\frac{\partial f}{\partial x_1} = 2x_1 = 0 \Rightarrow x_1 = 0$$

$$\frac{\partial f}{\partial x_2} = x_2 = 0 \Rightarrow x_2 = 0$$

$\Rightarrow f = 3$

Minimum of f is 3 at $(0, 0)$

Q2.2

each dimension has to have a representation of 32 bit, so I construct a matrix of 64 colomns to represent x1 and x2.

```
# Generate random binary matrix(Initial population)
initCrom = function(n){
  mat = matrix(NA, nrow = n, ncol = 64)

  for (i in 1:n) {
    u = runif(64, 0, 1)
    mat[i,] = ifelse(u < 0.5, 0, 1)
  }
  return(mat)

}

#v=initCrom(8)
#v


# Function to transform a binary vector to the integer it represents
binToint = function(v){
  y= Reduce(function(s,r) {s*2+r}, v)

  return(y)
}
#v = c(1,0,1)
#binToint(v)


#Function to represent the individual in logarithmic system
logScale = function(v){
  y1 = binToint(v[3:32])

  y2 = binToint(v[35:64])
  (-1)^v[1] * exp(y1*(-1)^v[2])
  x1 = (-1)^v[1] * exp(y1*(-1)^v[2])
  x2 = (-1)^v[33] * exp(y2*(-1)^v[34])
  # z1 = log(x1)
  # z2 = log(x2)
  return(c(x1, x2))
}

#logScale(v[5,])


#Function for evaluating the values of individuals
evalCrom1 = function(v){
  x1 = logScale(v)[1]
  x2 = logScale(v)[2]


  value = f(x1,x2)
```

```
  return(value)
}

# #Function for evaluating the values of individuals
evalCrom2 = function(v){
  x1 = binToint(v[1:32])
  x2 = binToint(v[1:32])


  value = f(x1,x2)

  return(value)
}

#evalCrom1(v[1,])
```

Q2.3

For this stage I decided to select 4 individuals randomly at two steps. Firs I randomle select 2 individuals, the I randomely select another two individuals. I evalute their values and choose the 2 fittest ones as the parents.

```
#with evalCrom1
Select1 = function(pop){
  N = nrow(pop)
  id = 1:N

  #Sample 2 individuals at random
  g1 = sample(id, 2)
  value1 = evalCrom1(pop[g1,])

  #Sample another 2 individuals at random
  g2 = sample(id[-g1], 2)
  value2 = evalCrom1(pop[g2,])

  best1 = g1[which.min(value1)]
  best2 = g2[which.min(value2)]
  #lose1 = g1[which.max(value1)]
  #lose2 = g2[which.max(value2)]
  select = matrix(c(pop[best1,], pop[best2,]),2,64)
  #pop = pop[-c(lose1,lose2),]

  return(select=select)
}

#with evalCrom2
Select2 = function(pop){
  N = nrow(pop)
  id = 1:N

  #Sample 2 individuals at random
  g1 = sample(id, 2)
  value1 = evalCrom2(pop[g1,])

  #Sample another 2 individuals at random
  g2 = sample(id[-g1], 2)
  value2 = evalCrom2(pop[g2,])

  best1 = g1[which.min(value1)]
  best2 = g2[which.min(value2)]
  #lose1 = g1[which.max(value1)]
  #lose2 = g2[which.max(value2)]
  select = matrix(c(pop[best1,], pop[best2,]),2,64)
  #pop = pop[-c(lose1,lose2),]

  return(select=select)
}

#v = initCrom(8)

#Select(v)
```

Q2.4

For crossover I decided to sample an integer between [1,32] and the swap over between two parents.

```
crossover = function(v1, v2){
  v1 = v1
  v2 = v2
  i = sample(1:32, 1)

  # j = sample(33:64, 1)
  #
  # k1 = v2[1:i]
  #
  # k2 = v1[i+1:32]
  # k3 = v2[33:j]
  # k4 = v1[j+1:64]
  v2[i:32] = v1[i:32]
  #kid = c(k1,k2,k3,k4)
  #offspring = c(v2[1:i],v1[(i+1):15])
  kid = matrix(v2, 1, 64)
  return(kid)
}
#kid = crossover(v[1,], v[2,])
#kid
```

For mutation I defined a "par" in argument which indicates howmany percent of the genes should be mutated. this parameter should'nt be large.

```
#Mutation
mutate = function(v, par){

  size = length(v)
  n = round(par * size)
  u = sample(1:size, n)
  for (i in 1:n) {
    id = u[i]
    v[id] = ifelse(v[id] == 0, 1, 0)

  }
  kid = matrix(v, 1, 64)

  return(kid)
}
```

```
genetic.func = function(maxiter, mutprob, initSize){


  #Initial population in binary
  initPop = initCrom(initSize)

  # # Function values of the initial population
  values = sapply(1:nrow(initPop), function(i){evalCrom1(initPop[i,])})


  Min = c()
  final.X = vector("list", length = maxiter)
```

```
  for (i in 1:maxiter) {

    # index with the maximum objective function
    victim.ind <- which.max(values)



    #selection from current population
    parent <- Select1(initPop)
    # produce new kid by crossovering the parents
    kid <- crossover(parent[1,], parent[2,])


    #new.kid = matrix(0, 1, 64)
    # Mutate this kid with probability mutprob
    if(runif(1) < mutprob){
      new.kid <- mutate(kid, 0.01)

    }else{

      new.kid <- kid[1,]

    }

    # Replacing victim with the kid
    initPop[victim.ind,] <- new.kid


    # Updating Values
    values <- sapply(1:nrow(initPop), function(i){evalCrom1(initPop[i,])})

    Min = c(Min, min(values))
    best.id = which.min(values)
    v = initPop[best.id]
    x = logScale(v)
    final.X[[i]] <- c(x[1],x[2])

  }
  #points(final.X, myFunc(final.X), col = "Red", lwd=3)
  selected = min(Min)
  id = which.min(Min)
  X = final.X[[id]]
  cat("Min value: ", selected, "\n")


}
```

When I use a logarithmic scale for individuals to be evaluated by the function value of the minimum is found:

```
genetic.func(100, 0.5,50)
```

```
## Min value:  3
```

However, when I use the integers themselves to be plugin the function the minimum could not be found. I probably have made a mistake in initializing a good population.

```
genetic.func = function(maxiter, mutprob, initSize){


  #Initial population in binary
  initPop = initCrom(initSize)

  # # Function values of the initial population
  values = sapply(1:nrow(initPop), function(i){evalCrom2(initPop[i,])})


  Min = c()
  final.X = vector("list", length = maxiter)

  for (i in 1:maxiter) {

    # index with the maximum objective function
    victim.ind <- which.max(values)



    #selection from current population
    parent <- Select2(initPop)
    # produce new kid by crossovering the parents
    kid <- crossover(parent[1,], parent[2,])


    #new.kid = matrix(0, 1, 64)
    # Mutate this kid with probability mutprob
    if(runif(1) < mutprob){
      new.kid <- mutate(kid, 0.01)

    }else{

      new.kid <- kid[1,]

    }

    # Replacing victim with the kid
    initPop[victim.ind,] <- new.kid


    # Updating Values
    values <- sapply(1:nrow(initPop), function(i){evalCrom2(initPop[i,])})

    Min = c(Min, min(values))
    best.id = which.min(values)
    v = initPop[best.id]
    x = logScale(v)
    final.X[[i]] <- c(x[1],x[2])

  }
```

```
  #points(final.X, myFunc(final.X), col = "Red", lwd=3)
  selected = min(Min)
  id = which.min(Min)
  X = final.X[[id]]
  cat("Min value: ", selected, "\n")


}
```

```
genetic.func(200, 0.5,50)
```

```
## Min value:  7.600183e+14
```

The problems with this implementation might be as follows:

- the representation of individuals have not been implemented properly
- choice of crossover and mutation is not good

Mutation probability should be as small as possible to prevent the algorithm to search just at random process. If we increase the mutation probability, we reduce the Genetic Algorithm to just a random search process. However, this cannot be shown properly by my algorithm.

Unfortunately I have no more time to work on the problems.